Filière IGE – 2017/2018
Algorithmique & Programmation
Leçon 5 : Tableaux

Prof. *A. DARGHAM*

Tableaux

- **Définitions :**
  - Un **tableau** est une **séquence** de N **objets** de **même type** et de **même nom**.
  - Les objets sont appelés les **éléments du tableau** et sont **numérotés consécutivement** de 0, à N-1.
  - Ces numéros sont les **indices** des éléments du tableau.
  - Si un tableau **T** de **N éléments**, alors **T[0]** est le nom de son 1er élément, **T[1]** est le nom de son 2ème élément, …, **T[N-1]** est le nom de son dernier élément.

Tableaux

- **Déclaration d'un tableau :**

  Type Nom [Nombre];

  - **Type** : est le **type des éléments** du tableau (***char***, ***int***, ***float***, ***double***, …).
  - **Nom** : est le **nom du tableau** (c'est un ***identificateur***).
  - **Nombre** : est le **nombre des éléments du tableau** (c'est une ***expression constante*** de ***type int***).

Tableaux

- **Exemples de déclaration de tableaux :**

```
char str[100];
int vecteur[3];
float notes[50];
double s[5];
#define N 10
int scores[N];
double poly[2*N+1];
int n = 10;
int v[n];  /* erreur : ce n'est pas une
expression constante */
```

Tableaux

- **Initialisation d'un tableau :**

  Type Nom [N] = {val_1 , val_2 , … , val_N} ;

  - **val_i** : est la **valeur du i^{ème} élément du tableau** de même type que **Type**.
  - Cette valeur doit être une **constante**.

Tableaux

- **Exemples d'initialisation de tableaux :**

```
int v[5] = {5, 7, 9, 11, 13};
float t[3] = {5.25, -0.5, 23.75};
int a[ ] = {2, 4, 6}; /* Ok, le compilateur va
  calculer le nombre des éléments à partir de
  la liste donnée */
int b[3] = {1, 2, 3, 4, 5}; /* Erreur */
int c[5] = {5, 7, 9}; /* Ok, le compilateur va
  compléter la liste par des 0 */
float x[5] = {5.5, 7.2}; /* Ok, le compilateur
  va compléter la liste par des 0.0 */
```

Tableaux

- **Exemples d'initialisation de tableaux :**

```
char str[6] = {'b', 'o', 'n', 'u', 's', '\0'};
char ch[7] = {'b', 'o', 'n'}; /* Ok, le
  compilateur va compléter la liste par des
  caractères '\0' */
```

Tableaux

- **L'opérateur & :**
  - L'opérateur **unaire *&*** est **l'opérateur d'adresse**.
  - Il ne s'applique que sur une variable.
  - L'expression « **&var** » donne **l'adresse de la variable** « **var** ».
  - L'adresse d'une variable est toujours un **nombre entier**.
  - Elle est ***calculée par le compilateur*** sous forme d'un ***entier en hexadécimal***.
  - Tous les arguments (sauf le premier) de la fonction **scanf** doivent être des adresses.

Tableaux

- **L'opérateur & :**
  - Pour **afficher l'adresse d'une variable** sous le **format *hexadécimal***, on utilise le spécificateur de format « **%p** » :

```
printf("Adresse de a : %p\n", &a);
```

  - Pour **afficher l'adresse d'une variable** sous le **format *décimal***, on utilise le spécificateur de format « **%d** » :

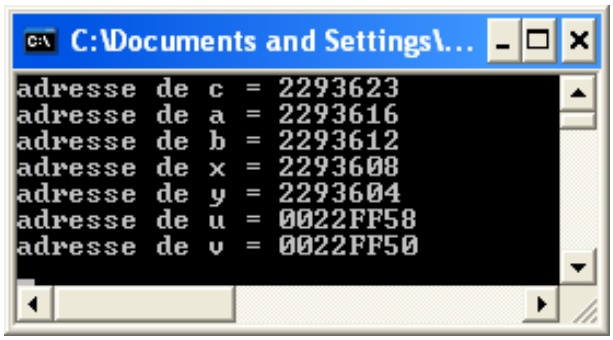
```
printf("Adresse de a : %d\n", &a);
```

Tableaux

- **Exemple d'utilisation de l'opérateur & :**

```
char c;
int a, b;
float x, y;
double u, v;
printf("adresse de c = %d\n", &c);
printf("adresse de a = %d\n", &a);
printf("adresse de b = %d\n", &b);
printf("adresse de x = %d\n", &x);
printf("adresse de y = %d\n", &y);
printf("adresse de u = %p\n", &u);
printf("adresse de v = %p\n", &v);
```

Tableaux



Tableaux

- **L'opérateur sizeof :**
  - L'opérateur **sizeof** donne **la taille mémoire** (c'est un **nombre entier mesuré** en **octets**) **d'un type de données** ou **d'une variable** selon la syntaxe suivante :

    sizeof(Type)

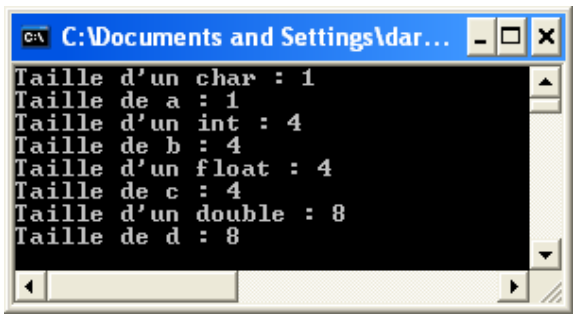
    sizeof(var)
  - La **taille mémoire** d'une variable est égale à **celle de son type**.

Tableaux

- Exemple 1 d'utilisation de l'opérateur sizeof :

```
char a;
int b;
float c;
double d;
printf("Taille d'un char : %d\n", sizeof(char));
printf("Taille de a : %d\n", sizeof(a));
printf("Taille d'un int : %d\n", sizeof(int));
printf("Taille de b : %d\n", sizeof(b));
printf("Taille d'un float : %d\n", sizeof(float));
printf("Taille de c : %d\n", sizeof(c));
printf("Taille d'un double : %d\n", sizeof(double));
printf("Taille de d : %d\n", sizeof(d));
```

Tableaux



Tableaux

- **Taille mémoire d'un tableau :**
  - Soient **A** un tableau de **N** éléments de type **T**. Alors :

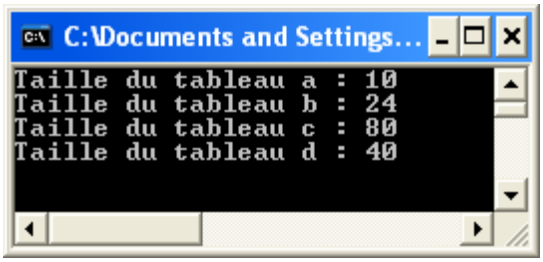
    sizeof(**A**) = **N** * **sizeof**(T)
  - La taille mémoire d'un tableau est le produit de son nombre d'éléments par la taille de son type.

Tableaux

- **Exemple 2 d'utilisation de l'opérateur sizeof :**

```
char a[10];
int b[6];
float c[20];
double d[5];
printf("Taille du tableau a : %d\n", sizeof(a));
printf("Taille du tableau b : %d\n", sizeof(b));
printf("Taille du tableau c : %d\n", sizeof(c));
printf("Taille du tableau d : %d\n", sizeof(d));
```

Tableaux



Tableaux

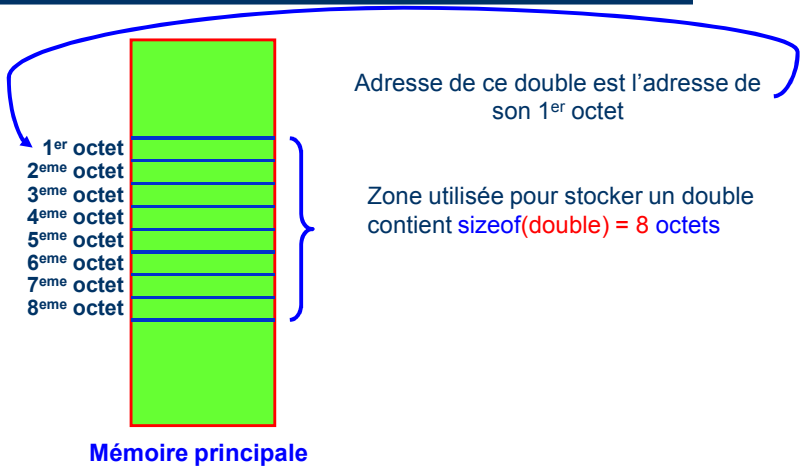
- **Remarques concernant l'adresse d'une variable :**
  - Soit **var** une variable de type **T**.
  - L'expression « **var** » désigne la valeur de la variable **var** : c'est une valeur de type **T**.
  - L'expression « **&var** » désigne l'adresse mémoire de la variable **var** : c'est un nombre entier.
  - L'expression « **sizeof(var)** » désigne la taille mémoire de la variable **var** : c'est un nombre entier.

Tableaux

- **Remarques concernant l'adresse d'une variable :**
  - Si **sizeof(var)** est égale à **k**, alors la variable **var** est rangée en mémoire dans un espace de **k octets consécutifs**.
  - La valeur de **&var** est égale à **l'adresse du premier octet** des **k octets réservés** pour la variable **var**.
  - La prochaine variable de type **T** juste après **var** aura une adresse égale à :

&var + k = &var + sizeof(var)

Tableaux

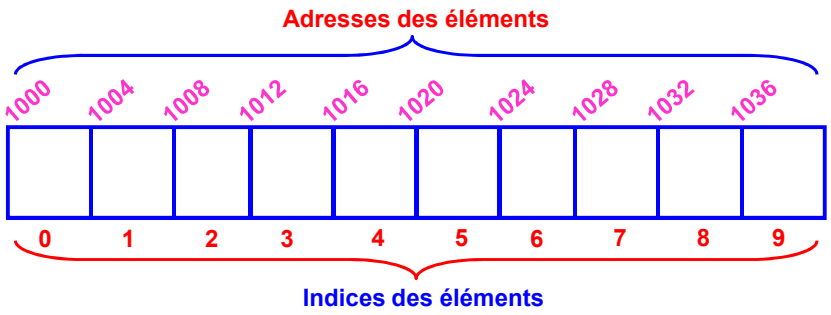


Tableaux

- **Représentation mémoire d'un tableau :**
  - Soient **A** un tableau de **N** éléments de type **T**.
  - Les **N** éléments de **A** sont rangés en mémoire dans un espace **contigu** (continu) :
  - **&A[1] = &A[0] + sizeof(T).**
  - **&A[2] = &A[1] + sizeof(T).**
  - **…**
  - **&A[N-1] = &A[N-2] + sizeof(T).**

Tableaux

- **Représentation mémoire d'un tableau :**



Représentation mémoire d'un tableau de 10 entiers. Le premier élément étant stocké à l'adresse 1000.

Tableaux

- **Manipulation des éléments d'un tableau :**
  - Soient **A** un tableau de **N** éléments de type **T**, et **i** un **entier** (une **expression quelconque**, mais **valide** de **type int**, et de **valeur comprise** entre **0** et **N-1**).
  - **A[i]** désigne **l'élément de A d'indice i**.
  - **A[i]** se manipule comme étant une **variable de type T** : ***tout ce qui est applicable sur une variable de type T, est aussi applicable sur A[i]***.

Tableaux

- **Exemples de manipulation des éléments d'un tableau :**

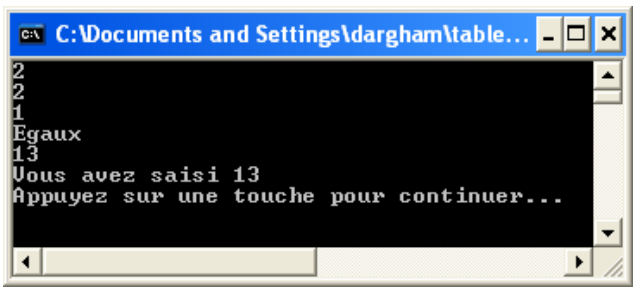
```
#include <stdio.h>
main()
{
   int a[5] = {2, 5, 8, 0, 2};
   /* afficher le premier élément de a */
   printf("%d\n", a[0]);
   /* afficher le dernier élément de a */
   printf("%d\n", a[4]);
   /* incrémenter l'avant dernier élément de a */
   a[3]++;
   printf("%d\n", a[3]);
```

Tableaux

- **Exemples de manipulation des éléments d'un tableau :**

```
    /* tester si le premier et le dernier éléments
       de a sont égaux */
    if(a[0] == a[4])  printf("Egaux\n");
    else printf("Differents\n");
    /* lire l'élément de a d'indice 2 */
    scanf("%d", &a[2]);
    printf("Vous avez saisi %d\n", a[]);
    system("pause");
}
```

Tableaux



Tableaux

- **Boucles et tableaux :**
  - Les **boucles** et les **tableaux** sont intimement liées.
  - Lorsqu'un **traitement doit être effectué à tous les éléments d'un tableau**, **l'utilisation des boucles s'impose**.
  - La boucle suivante permet de **lire et d'afficher tous les éléments d'un tableau T** ayant **N** éléments de type **int** :

Tableaux

- **Exemple :**

```
#include <stdio.h>
main()
{
   int a[5];
   int i; /* indice */
   printf("Lecture des elements de a:\n");
   for(i = 0; i <= 4; i++)
   {
      printf("a[%d] = ", i);
      scanf("%d", &a[i]);
   }
```

Tableaux

```
    printf("Affichage des elements de a:\n");
    for(i = 0; i <= 4; i++)
       printf("%d\t", a[i]);
    printf("\n");
    system("pause");
}
```

Tableaux